

# Introduction to Computational Linguistics

Pavlina Ivanova

University of Plovdiv, Bulgaria

## **Lecture 3: Part-of-Speech Tagging**

Thanks to Daniel Jurafsky for these slides

# Part of Speech tagging

- Part of speech tagging
  - Parts of speech
  - What's POS tagging good for anyhow?
  - Tag sets
  - Rule-based tagging
  - Statistical tagging
    - Simple most-frequent-tag baseline
  - Important Ideas
    - Training sets and test sets
    - Unknown words
  - TB tagging
  - HMM tagging

# Parts of Speech

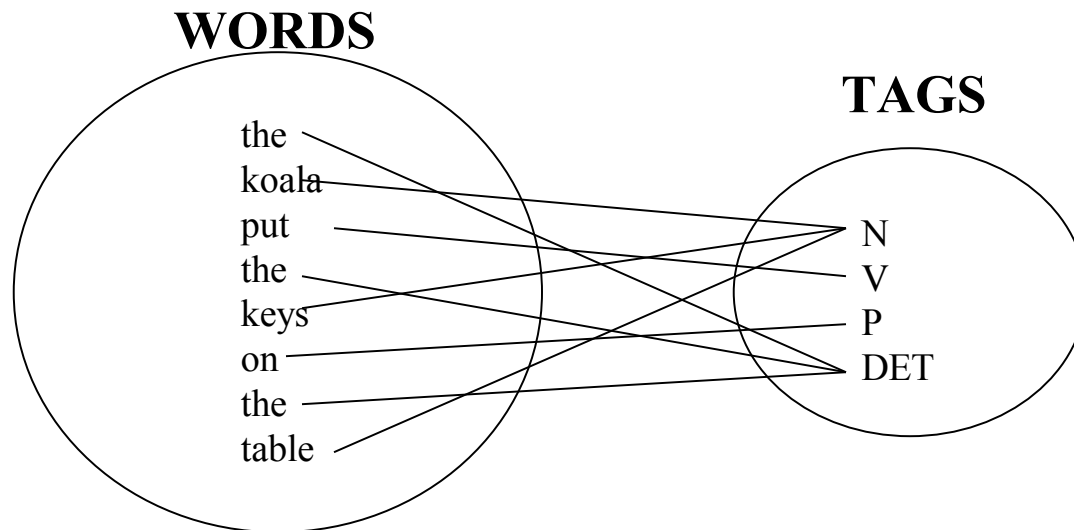
- 8 traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, pronoun, conjunction
  - This idea has been around for over 2000 years (Dionysius Thrax of Alexandria, c. 100 B.C.)
  - Called: parts-of-speech, lexical categories, word classes, morphological classes, lexical tags, POS
  - Lots of debate in linguistics about the number, nature, and universality of these
    - We'll completely ignore this debate.

# POS examples

- N            noun            chair, bandwidth, pacing
- V            verb            study, debate, munch
- ADJ        adjective        purple, tall, ridiculous
- ADV        adverb        unfortunately, slowly
- P            preposition    of, by, to
- PRO        pronoun        I, me, mine
- DET        determiner    the, a, that, those

# POS Tagging: Definition

- The process of assigning a part-of-speech or lexical class marker to each word in a text/corpus:



# POS Tagging example

WORD	tag
<b>the</b>	<b>DET</b>
<b>koala</b>	<b>N</b>
<b>put</b>	<b>V</b>
<b>the</b>	<b>DET</b>
<b>keys</b>	<b>N</b>
<b>on</b>	<b>P</b>
<b>the</b>	<b>DET</b>
<b>table</b>	<b>N</b>

# What is POS tagging good for?

- Is the first step of a vast number of Comp Ling tasks
- Speech synthesis:
  - How to pronounce “lead”?
  - INsult                      inSULT
  - OBject                      obJECT
  - OVERflow                      overFLOW
  - DIScount                      disCOUNT
  - CONtent                      conTENT
- Parsing
  - Need to know if a word is an N or V before you can parse
- Word prediction in speech recognition and etc
  - Possessive pronouns (my, your, her) followed by nouns
  - Personal pronouns (I, you, he) likely to be followed by verbs
- Machine Translation, etc

# Open and closed class words

- Closed class: a relatively fixed membership
  - Prepositions: of, in, by, ...
  - Auxiliaries: may, can, will had, been, ...
  - Pronouns: I, you, she, mine, his, them, ...
  - Usually **function words** (short common words which play a role in grammar)
- Open class: new ones can be created all the time
  - English has 4: Nouns, Verbs, Adjectives, Adverbs
  - Many languages have all 4, but not all!
  - In Lakota and possibly Chinese, what English treats as adjectives act more like verbs.



# Open class words

- Nouns
  - Proper nouns (Stanford University, Boulder, Neal Snider, Margaret Jacks Hall). English capitalizes these.
  - Common nouns (the rest). German capitalizes these.
  - Count nouns and mass nouns
    - Count: have plurals, get counted: goat/goats, one goat, two goats
    - Mass: don't get counted (snow, salt, communism) (\*two snows)
- Adverbs: tend to modify things
  - Unfortunately, John walked home extremely slowly yesterday
  - Directional/locative adverbs (here, home, downhill)
  - Degree adverbs (extremely, very, somewhat)
  - Manner adverbs (slowly, slinkily, delicately)
- Verbs:
  - In English, have morphological affixes (eat/eats/eaten)

# Closed Class Words

- Idiosyncratic
- Examples:
  - prepositions: on, under, over, ...
  - particles: up, down, on, off, ...
  - determiners: a, an, the, ...
  - pronouns: she, who, I, ..
  - conjunctions: and, but, or, ...
  - auxiliary verbs: can, may should, ...
  - numerals: one, two, three, third, ...

# Prepositions from CELEX

of	540,085	through	14,964	worth	1,563	pace	12
in	331,235	after	13,670	toward	1,390	nigh	9
for	142,421	between	13,275	plus	750	re	4
to	125,691	under	9,525	till	686	mid	3
with	124,965	per	6,515	amongst	525	o'er	2
on	109,129	among	5,090	via	351	but	0
at	100,169	within	5,030	amid	222	ere	0
by	77,794	towards	4,700	underneath	164	less	0
from	74,843	above	3,056	versus	113	midst	0
about	38,428	near	2,026	amidst	67	o'	0
than	20,210	off	1,695	sans	20	thru	0
over	18,071	past	1,575	circa	14	vice	0

# English particles

aboard	aside	besides	forward(s)	opposite	through
about	astray	between	home	out	throughout
above	away	beyond	in	outside	together
across	back	by	inside	over	under
ahead	before	close	instead	overhead	underneath
alongside	behind	down	near	past	up
apart	below	east, etc.	off	round	within
around	beneath	eastward(s),etc.	on	since	without

# Pronouns: CELEX

it	199,920	how	13,137	yourself	2,437	no one	106
I	198,139	another	12,551	why	2,220	wherein	58
he	158,366	where	11,857	little	2,089	double	39
you	128,688	same	11,841	none	1,992	thine	30
his	99,820	something	11,754	nobody	1,684	summat	22
they	88,416	each	11,320	further	1,666	suchlike	18
this	84,927	both	10,930	everybody	1,474	fewest	15
that	82,603	last	10,816	ourselves	1,428	thyslf	14
she	73,966	every	9,788	mine	1,426	whomever	11
her	69,004	himself	9,113	somebody	1,322	whosoever	10
we	64,846	nothing	9,026	former	1,177	whomsoever	8
all	61,767	when	8,336	past	984	wherefore	6
which	61,399	one	7,423	plenty	940	whereat	5
their	51,922	much	7,237	either	848	whatsoever	4
what	50,116	anything	6,937	yours	826	whereon	2
my	46,791	next	6,047	neither	618	whoso	2
him	45,024	themselves	5,990	fewer	536	aught	1
me	43,071	most	5,115	hers	482	howsoever	1
who	42,881	itself	5,032	ours	458	thrice	1
them	42,099	myself	4,819	whoever	391	wheresoever	1
no	33,458	everything	4,662	least	386	you-all	1
some	32,863	several	4,306	twice	382	additional	0
other	29,391	less	4,278	theirs	303	anybody	0
your	28,923	herself	4,016	wherever	289	each other	0
its	27,783	whose	4,005	oneself	239	once	0
our	23,029	someone	3,755	thou	229	one another	0
these	22,697	certain	3,345	'un	227	overmuch	0
any	22,666	anyone	3,318	ye	192	such and such	0
more	21,873	whom	3,229	thy	191	whate'er	0
many	17,343	enough	3,197	whereby	176	whenever	0
such	16,880	half	3,065	thee	166	whereof	0
those	15,819	few	2,933	yourselves	148	whereto	0
own	15,741	everyone	2,812	latter	142	whereunto	0
us	15,724	whatever	2,571	whichever	121	whichsoever	0

# Conjunctions

and	514,946	yet	5,040	considering	174	forasmuch as	0
that	134,773	since	4,843	lest	131	however	0
but	96,889	where	3,952	albeit	104	immediately	0
or	76,563	nor	3,078	providing	96	in as far as	0
as	54,608	once	2,826	whereupon	85	in so far as	0
if	53,917	unless	2,205	seeing	63	inasmuch as	0
when	37,975	why	1,333	directly	26	insomuch as	0
because	23,626	now	1,290	ere	12	insomuch that	0
so	12,933	neither	1,120	notwithstanding	3	like	0
before	10,720	whenever	913	according as	0	neither nor	0
though	10,329	whereas	867	as if	0	now that	0
than	9,511	except	864	as long as	0	only	0
while	8,144	till	686	as though	0	provided that	0
after	7,042	provided	594	both and	0	providing that	0
whether	5,978	whilst	351	but that	0	seeing as	0
for	5,935	suppose	281	but then	0	seeing as how	0
although	5,424	cos	188	but then again	0	seeing that	0
until	5,072	supposing	185	either or	0	without	0

# POS tagging: Choosing a tagset

- There are so many parts of speech, potential distinctions we can draw
- To do POS tagging, need to choose a standard set of tags to work with
- Could pick very coarse tagsets
  - N, V, Adj, Adv.
- More commonly used set is finer grained, the “UPenn TreeBank tagset”, 45 tags
  - PRP\$, WRB, WP\$, VBG
- Even more fine-grained tagsets exist

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	<i>and, but, or</i>	SYM	Symbol	<i>+, %, &amp;</i>
CD	Cardinal number	<i>one, two, three</i>	TO	"to"	<i>to</i>
DT	Determiner	<i>a, the</i>	UH	Interjection	<i>ah, oops</i>
EX	Existential "there"	<i>there</i>	VB	Verb, base form	<i>eat</i>
FW	Foreign word	<i>mea culpa</i>	VBD	Verb, past tense	<i>ate</i>
IN	Preposition/sub-conj	<i>of, in, by</i>	VBG	Verb, gerund	<i>eating</i>
JJ	Adjective	<i>yellow</i>	VBN	Verb, past participle	<i>eaten</i>
JJR	Adj., comparative	<i>btgger</i>	VBP	Verb, non-3sg pres	<i>eat</i>
JJS	Adj., superlative	<i>wildest</i>	VBZ	Verb, 3sg pres	<i>eats</i>
LS	List item marker	<i>1, 2, One</i>	WDT	Wh-determiner	<i>whtch, that</i>
MD	Modal	<i>can, should</i>	WP	Wh-pronoun	<i>what, who</i>
NN	Noun, sing. or mass	<i>llama</i>	WPS	Possessive wh-	<i>whose</i>
NNS	Noun, plural	<i>llamas</i>	WRB	Wh-adverb	<i>how, where</i>
NNP	Proper noun, singular	<i>IBM</i>	\$	Dollar sign	<i>\$</i>
NNPS	Proper noun, plural	<i>Carolmas</i>	#	Pound sign	<i>#</i>
PDT	Predeterminer	<i>all, both</i>	"	Left quote	<i>(" or ")</i>
POS	Possessive ending	<i>'s</i>	"	Right quote	<i>(' or ')</i>
<b>PRP</b>	Personal pronoun	<i>I, you, he</i>	(	Left parenthesis	<i>( [ ( { &lt;</i>
<b>PRP\$</b>	Possessive pronoun	<i>your, one's</i>	)	Right parenthesis	<i>) ] } &gt;</i>
RB	Adverb	<i>quickly, never</i>	,	Comma	<i>,</i>
RBR	Adverb, comparative	<i>faster</i>	.	Sentence-final punc	<i>. ! ?</i>
RBS	Adverb, superlative	<i>fastest</i>	:	Mid-sentence punc	<i>: ; ... - -</i>
RP	Particle	<i>up, off</i>			



# Using the UPenn tagset

- The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.
- Prepositions and subordinating conjunctions marked IN (“although/IN I/PRP..”)
- Except the preposition/complementizer “to” is just marked “to”.

# POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- **The POS tagging problem is to determine the POS tag for a particular instance of a word.**

# How hard is POS tagging? Measuring ambiguity

	Original 87-tag corpus	Trebank 45-tag corpus
<b>Unambiguous (1 tag)</b>	<b>44,019</b>	<b>38,857</b>
<b>Ambiguous (2–7 tags)</b>	<b>5,490</b>	<b>8844</b>
Details:		
2 tags	4,967	6,731
3 tags	411	1621
4 tags	91	357
5 tags	17	90
6 tags	2 ( <i>well, beat</i> )	32
7 tags	2 ( <i>still, down</i> )	6 ( <i>well, set, round, open, fit, down</i> )
8 tags		4 ( <i>'s, half, back, a</i> )
9 tags		3 ( <i>that, more, in</i> )

# 3 methods for POS tagging

1. Rule-based tagging
  - (ENGTWOL)
2. Stochastic (=Probabilistic) tagging
  - HMM (Hidden Markov Model) tagging
3. Transformation-based tagging
  - Brill tagger

# Rule-based tagging

- Start with a dictionary
- Assign all possible tags to words from the dictionary
- Write rules by hand to selectively remove tags
- Leaving the correct tag for each word.

# Start with a dictionary

- she: PRP
- promised: VBN, VBD
- to TO
- back: VB, JJ, RB, NN
- the: DT
- bill: NN, VB
- Etc... for the ~100,000 words of English

Use the dictionary to assign every possible tag

			NN			
			RB			
	VBN		JJ			VB
PRP	VBD	TO	VB	DT	NN	
<b>She</b>	<b>promised</b>	<b>to</b>	<b>back</b>		<b>the</b>	<b>bill</b>

# Write rules to eliminate tags

Eliminate VBN if VBD is an option when VBN|  
VBD follows “<start> PRP”

			NN			
			RB			
	<b>VBN</b>		JJ			VB
PRP	VBD	TO	VB	DT	NN	
<b>She</b>	<b>promised</b>	<b>to</b>	<b>back</b>		<b>the</b>	<b>bill</b>



# Sample ENGTWOL Lexicon

<b>Word</b>	<b>POS</b>	<b>Additional POS features</b>
smaller	ADJ	COMPARATIVE
entire	ADJ	ABSOLUTE ATTRIBUTIVE
fast	ADV	SUPERLATIVE
that	DET	CENTRAL DEMONSTRATIVE SG
all	DET	PREDETERMINER SG/PL QUANTIFIER
dog's	N	GENITIVE SG
furniture	N	NOMINATIVE SG NOINDEFDETERMINE R
one-third	NUM	SG
she	PRON	PERSONAL FEMININE NOMINATIVE SG3
show	V	IMPERATIVE VFIN
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

# Stage 1 of ENGTWOL Tagging

- First Stage: Run words through FST morphological analyzer to get all parts of speech.

- Example: *Pavlov had shown that salivation ...*

Pavlov      **PAVLOV N NOM SG PROPER**

had          **HAVE V PAST VFIN SVO**

**HAVE PCP2 SVO**

shown      **SHOW PCP2 SVOO SVO SV**

that        **ADV**

**PRON DEM SG**

**DET CENTRAL DEM SG**

**CS**

salivation **N NOM SG**

# Stage 2 of ENGTWOL Tagging

- Second Stage: Apply NEGATIVE constraints.
- Example: Adverbial “that” rule
  - Eliminates all readings of “that” except the one in
    - “It isn’t *that* odd”

**Given** input: “that”

**If**

(+1 A/ADV/QUANT) ;if next word is adj/adv/quantifier

(+2 SENT-LIM) ;following which is E-O-S

(NOT -1 SVOC/A) ; and the previous word is not a

; verb like “consider” which

; allows adjective complements

; in “I consider that odd”

**Then** eliminate non-ADV tags

**Else** eliminate ADV

# Statistical Tagging

- Based on probability theory
- First we'll introduce the simple “most-frequent-tag” algorithm
- **Most-freq-tag** is another **baseline** algorithm.
- Meaning that no one would use it if they really wanted some data tagged
- But it's useful as a comparison

# Conditional Probability and Tags

- $P(\text{Verb})$  is the probability of a randomly selected word being a verb.
- $P(\text{Verb}|\text{race})$  is “what’s the probability of a word being a verb given that it’s the word ‘race’”?
- Race can be a noun or a verb.
- It’s more likely to be a noun.
- $P(\text{Verb}|\text{race})$  can be estimated by looking at some corpus and saying “out of all the times we saw ‘race’, how many were verbs?”

$$P(V | \text{race}) = \frac{\text{Count}(\text{race is verb})}{\text{total Count}(\text{race})}$$

- In Brown corpus,  $P(\text{Verb}|\text{race}) = 2/98 = .02$

# Most frequent tag

- Some ambiguous words have a more frequent tag and a less frequent tag:
- Consider the word “a” in these 2 sentences:
  - would/MD prohibit/VB a/DT suit/NN for/IN refund/NN
  - of/IN section/NN 381/CD (( a/NN ))/. ./.
- Which do you think is more frequent?

# Counting in a corpus

- We could count in a corpus
- A corpus: an on-line collection of text, often linguistically annotated
- The Brown Corpus: 1 million words from 1961
- Part of speech tagged at U Penn
- I counted in this corpus
- The results:

21830	DT
6	NN
3	FW

# The Most Frequent Tag algorithm

- For each word
  - Create a dictionary with each possible tag for a word
  - Take a tagged corpus
  - Count the number of times each tag occurs for that word
- Given a new sentence
  - For each word, pick the most frequent tag for that word from the corpus.



# The Most Frequent Tag algorithm: the dictionary

- For each word, we said:
  - Create a dictionary with each possible tag for a word...
- Q: Where does the dictionary come from?
- A: One option is to use the same corpus that we use for computing the tags

# Using a corpus to build a dictionary

- The/DT City/NNP Purchasing/NNP Department/NNP ,/, the/DT jury/NN said/VBD,/, is/VBZ lacking/VBG in/IN experienced/VBN clerical/JJ personnel/NNS ...
- From this sentence, dictionary is:
  - clerical
  - department
  - experienced
  - in
  - is
  - jury
  - ...

# Evaluating performance

- How do we know how well a tagger does?
- Say we had a test sentence, or a set of test sentences, that were already tagged by a human (a “Gold Standard”)
- We could run a tagger on this set of test sentences
- And see how many of the tags we got right.
- This is called “Tag accuracy” or “Tag percent correct”

# Test set

- We take a set of test sentences
- Hand-label them for part of speech
- The result is a “Gold Standard” test set
- Who does this?
  - Brown corpus: done by U Penn
  - Grad students in linguistics
- Don't they disagree?
  - Yes! But on about 97% of tags no disagreements
  - And if you let the taggers discuss the remaining 3%, they often reach agreement

# Training and test sets

- But we can't train our frequencies on the test set sentences.
- So for testing the Most-Frequent-Tag algorithm (or any other stochastic algorithm), we need 2 things:
  - A hand-labeled training set: the data that we compute frequencies from, etc
  - A hand-labeled test set: The data that we use to compute our % correct.

# Computing % correct

- Of all the words in the test set
- For what percent of them did the tag chosen by the tagger equal the human-selected tag.

$$\% \text{ correct} = \frac{\text{\# of words tagged correctly in test set}}{\text{total \# of words in test set}}$$

- Human tag set: (“Gold Standard” set)

# Training and Test sets

- Often they come from the same labeled corpus!
- We just use 90% of the corpus for training and save out 10% for testing!
- Even better: cross-validation
  - Take 90% training, 10% test, get a % correct
  - Now take a different 10% test, 90% training, get % correct
  - Do this 10 times and average

# Evaluation and rule-based taggers

- Does the same evaluation metric work for rule-based taggers?
- Yes!
  - Rule-based taggers don't need the training set.
  - But they still need a test set to see how well the rules are working.



# Unknown Words

- Most-frequent-tag approach has a problem!!
- What about words that don't appear in the training set?
- For example, here are some words that occur in a small Brown Corpus test set but not the training set:
  - Abernathy                      azalea                      alligator
  - absolution                      baby-sitter                asparagus
  - Adrien                            bantered boxcar
  - ajar                                bare-armed                boxcars
  - Alicia                             big-boned                 bumped
  - all-american-boy                boathouses

# Unknown words

- New words added to (newspaper) language 20+ per month
- Plus many proper names ...
- Increases error rates by 1-2%
- Method 1: assume they are nouns
- Method 2: assume the unknown words have a probability distribution similar to words only occurring once in the training set.
- Method 3: Use morphological information, e.g., words ending with -ed tend to be tagged VBN.

# Transformation-Based Tagging (Brill Tagging)

- Combination of Rule-based and stochastic tagging methodologies
  - Like rule-based because rules are used to specify tags in a certain environment
  - Like stochastic approach because machine learning is used—with tagged corpus as input
- Input:
  - tagged corpus
  - dictionary (with most frequent tags)

# Transformation-Based Tagging (cont.)

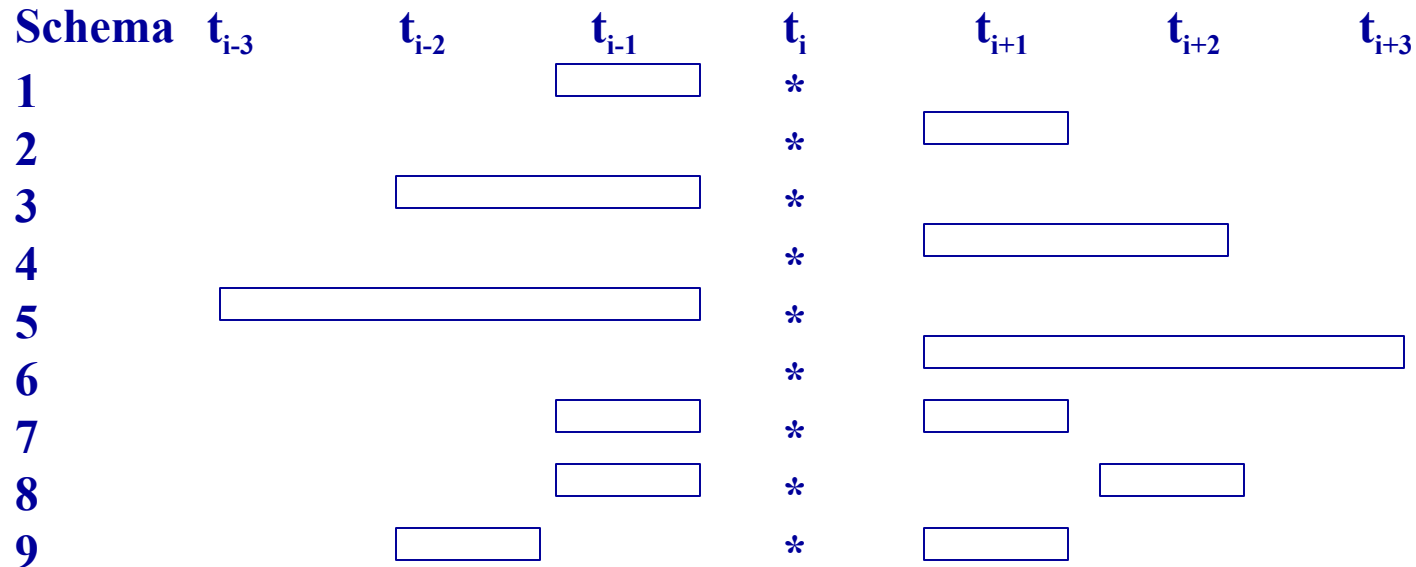
- Basic Idea:
  - Set the most probable tag for each word as a start value
  - Change tags according to rules of type “if word-1 is a determiner and word is a verb then change the tag to noun” in a specific order
- Training is done on tagged corpus:
  - Write a set of rule templates
  - Among the set of rules, find one with highest score
  - Continue from 2 until lowest score threshold is passed
  - Keep the ordered set of rules
- Rules make errors that are corrected by later rules

# TBL Rule Application

- Tagger labels every word with its most-likely tag
  - For example: *race* has the following probabilities in the Brown corpus:
    - $P(NN|race) = .98$
    - $P(VB|race) = .02$
- Transformation rules make changes to tags
  - “Change NN to VB when previous tag is TO”  
... *is/VBZ expected/VBN to/TO race/NN tomorrow/NN*  
becomes  
... *is/VBZ expected/VBN to/TO race/VB tomorrow/NN*

# TBL: Rule Learning

- 2 parts to a rule
  - Triggering environment
  - Rewrite rule
- The range of triggering environments of templates



# TBL: The Tagging Algorithm

- Step 1: Label every word with most likely tag (from dictionary)
- Step 2: Check every possible transformation & select one which most improves tagging
- Step 3: Re-tag corpus applying the rules
- Repeat 2-3 until some criterion is reached, e.g.,  $X\%$  correct with respect to training corpus
- **RESULT:** Sequence of transformation rules

# TBL: Rule Learning (cont.)

- Problem: Could apply transformations ad infinitum!
- Constrain the set of transformations with “templates”:
  - Replace tag  $X$  with tag  $Y$ , provided tag  $Z$  or word  $Z$  appears in some position
- Rules are learned in ordered sequence
- Rules may interact.
- Rules are compact and can be inspected by humans



# Templates for TBL

The preceding (following) word is tagged **z**.

The word two before (after) is tagged **z**.

One of the two preceding (following) words is tagged **z**.

One of the three preceding (following) words is tagged **z**.

The preceding word is tagged **z** and the following word is tagged **w**.

The preceding (following) word is tagged **z** and the word  
two before (after) is tagged **w**.

#	Change tags		Condition	Example
	From	To		
1	NN	VB	Previous tag is TO	to/TO race/NN → VB
2	VBP	VB	One of the previous 3 tags is MD	might/MD vanish/VBP → VB
3	NN	VB	One of the previous 2 tags is MD	might/MD not reply/NN → VB
4	VB	NN	One of the previous 2 tags is DT	
5	VBD	VBN	One of the previous 3 tags is VBZ	

# Isolated word error correction

- Suppose the user types “**graffe**” and we want to correct it
- How do I fix “**graffe**”?
  - Search through all words:
    - graf
    - craft
    - grail
    - giraffe
  - Pick the one that’s closest to **graffe**
  - What does “closest” mean?
  - We need a **distance metric**.
  - The simplest one: **edit distance**.
    - (More sophisticated probabilistic ones: noisy channel)

# Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
  - Insertion
  - Deletion
  - Substitution
- Needed to transform one into the other

# Minimum Edit Distance

I N T E \* N T I O N  
| | | | | | | | |  
\* E X E C U T I O N  
d s s i s


- If each operation has cost of 1
- Distance between these is 5
- If substitutions cost 2 (Levenshtein)
- Distance between these is 8

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	<b>0</b>	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

N	9									
O	8									
I	7									
T	3									

$$distance[i, j] = \min \begin{cases} distance[i-1, j] + ins-cost(target_{i-1}) \\ distance[i-1, j-1] + subst-cost(source_{j-1}, target_{i-1}) \\ distance[i, j-1] + del-cost(source_{j-1}) \end{cases}$$

T	3									
N	2									
I	1									
#	<b>0</b>	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N



N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

# Suppose we want the alignment too

- We can keep a “backtrace”
- Every time we enter a cell, remember where we came from
- Then when we reach the end, we can trace back from the upper right corner to get an alignment



N	9	8	9	10	11	12	11	10	9	<b>8</b>
O	8	7	8	9	10	11	10	9	<b>8</b>	9
I	7	6	7	8	9	10	9	<b>8</b>	9	10
T	6	5	6	7	8	9	<b>8</b>	9	10	11
N	5	4	5	6	7	<b>8</b>	9	10	11	10
E	4	3	4	<b>5</b>	<b>6</b>	<b>7</b>	8	9	10	9
T	3	4	<b>5</b>	6	7	8	7	8	9	8
N	2	<b>3</b>	4	5	6	7	8	7	8	7
I	<b>1</b>	2	3	4	5	6	7	6	7	8
#	<b>0</b>	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

**function** MIN-EDIT-DISTANCE(*target*, *source*) **returns** *min-distance*

$n \leftarrow \text{LENGTH}(\textit{target})$

$m \leftarrow \text{LENGTH}(\textit{source})$

Create a distance matrix  $\textit{distance}[n+1, m+1]$

Initialize the zeroth row and column to be the distance from the empty string

**for each column**  $i$  **from** 0 **to**  $n$  **do**

$\textit{distance}[i, 0] \leftarrow i$

**for each row**  $j$  **from** 0 **to**  $m$  **do**

$\textit{distance}[0, j] \leftarrow j$

**for each column**  $i$  **from** 1 **to**  $n$  **do**

**for each row**  $j$  **from** 1 **to**  $m$  **do**

$\textit{distance}[i, j] \leftarrow \text{MIN}(\textit{distance}[i-1, j] + \textit{ins-cost}(\textit{target}_{i-1}),$   
 $\textit{distance}[i-1, j-1] + \textit{subst-cost}(\textit{source}_{j-1}, \textit{target}_{i-1}),$   
 $\textit{distance}[i, j-1] + \textit{del-cost}(\textit{source}_{j-1}))$

**Figure 3.25** The minimum edit distance algorithm, an example of the class of dynamic programming algorithms.

# Summary

- Minimum Edit Distance
- A “dynamic programming” algorithm
- A probabilistic version of this called “Viterbi” is a key part of the Hidden Markov Model!

# Hidden Markov Model Tagging

- Using an HMM to do POS tagging
- Is a special case of Bayesian inference
  - Foundational work in computational linguistics
  - Bledsoe 1959: OCR
  - Mosteller and Wallace 1964: authorship identification
- It is also related to the “noisy channel” model

# POS tagging as a sequence classification task

- We are given a sentence (an “observation” or “sequence of observations”)
  - Secretariat is expected to race tomorrow
- What is the best sequence of tags which corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of  $n$  words  $w_1 \dots w_n$ .

# Getting to HMM

- We want, out of all sequences of  $n$  tags  $t_1 \dots t_n$  the single tag sequence such that  $P(t_1 \dots t_n | w_1 \dots w_n)$  is highest

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

- Hat  $\hat{\phantom{x}}$  means “our estimate of the best one”
- $\operatorname{Argmax}_x f(x)$  means “the  $x$  such that  $f(x)$  is maximized”

# Getting to HMM

- This equation is guaranteed to give us the best tag sequence

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

- But how to make it operational? How to compute this value?
- Intuition of Bayesian classification:
  - Use Bayes rule to transform into a set of other probabilities that are easier to compute

# Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(w_1^n | t_1^n) P(t_1^n)$$



# Likelihood and prior

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} \overbrace{P(w_1^n | t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^n P(w_i | t_i)$$

$$P(t_1^n) \approx \prod_{i=1}^n P(t_i | t_{i-1})$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$

# Two kinds of probabilities (1)

- Tag transition probabilities  $p(t_i|t_{i-1})$ 
  - Determiners likely to precede adjs and nouns
    - That/DT flight/NN
    - The/DT yellow/JJ hat/NN
    - So we expect  $P(NN|DT)$  and  $P(JJ|DT)$  to be high
  - Compute  $P(NN|DT)$  by counting in a labeled corpus:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

# Two kinds of probabilities (2)

- Word likelihood probabilities  $p(w_i|t_i)$ 
  - VBZ (3sg Pres verb) likely to be “is”
  - Compute  $P(\text{is}|\text{VBZ})$  by counting in a labeled corpus:

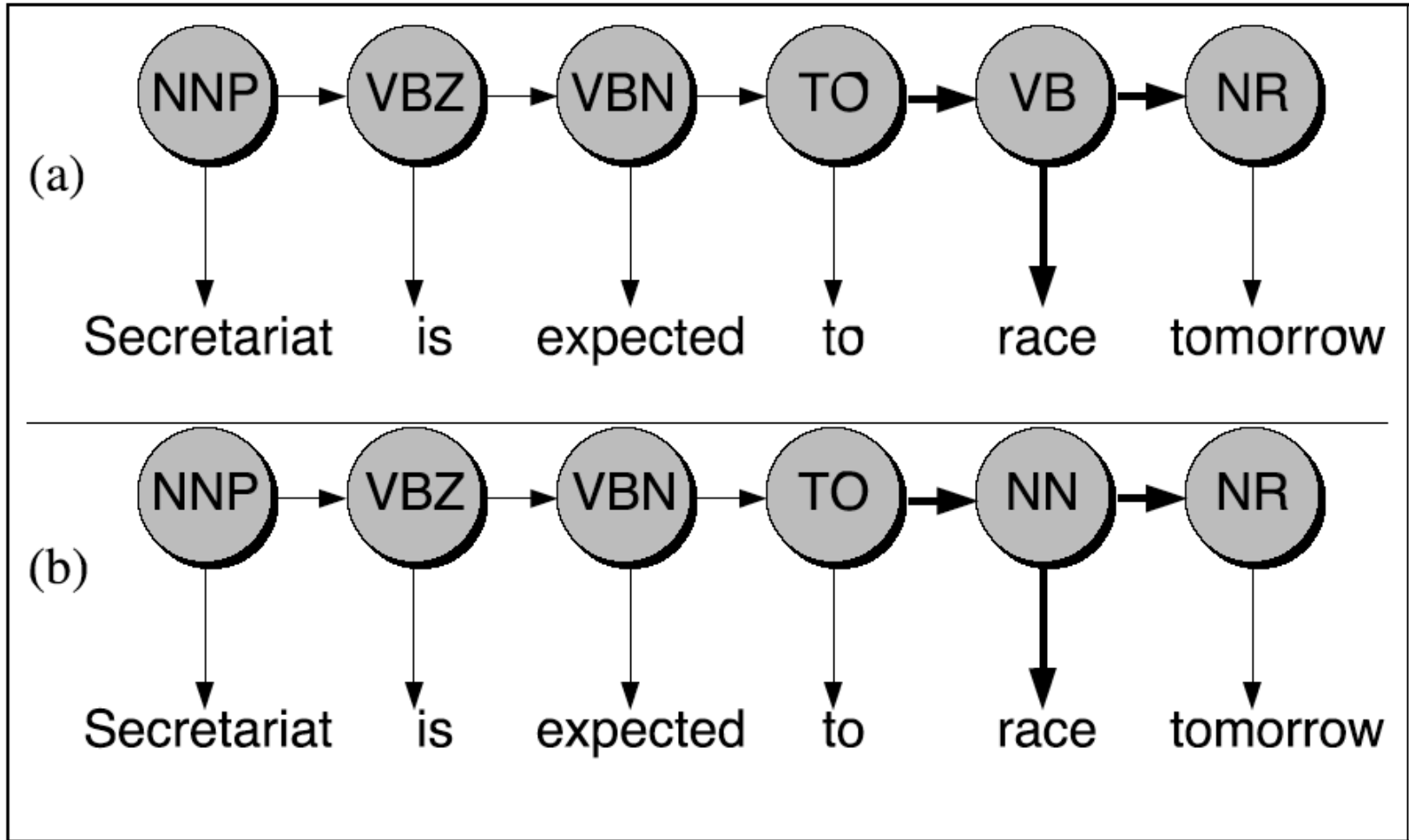
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(\text{is}|\text{VBZ}) = \frac{C(\text{VBZ}, \text{is})}{C(\text{VBZ})} = \frac{10,073}{21,627} = .47$$

# An Example: the verb “race”

- Secretariat/**NNP** is/**VBZ** expected/**VCN** to/**TO** **race**/**VB** tomorrow/**NR**
- People/**NNS** continue/**VB** to/**TO** inquire/**VB** the/**DT** reason/**NN** for/**IN** the/**DT** **race**/**NN** for/**IN** outer/**JJ** space/**NN**
- How do we pick the right tag?

# Disambiguating “race”



- $P(\text{NN}|\text{TO}) = .00047$
- $P(\text{VB}|\text{TO}) = .83$
- $P(\text{race}|\text{NN}) = .00057$
- $P(\text{race}|\text{VB}) = .00012$
- $P(\text{NR}|\text{VB}) = .0027$
- $P(\text{NR}|\text{NN}) = .0012$
- $P(\text{VB}|\text{TO})P(\text{NR}|\text{VB})P(\text{race}|\text{VB}) = .00000027$
- $P(\text{NN}|\text{TO})P(\text{NR}|\text{NN})P(\text{race}|\text{NN}) = .00000000032$
- So we (correctly) chose the verb reading,

# Hidden Markov Models

- What we've described with these two kinds of probabilities is a Hidden Markov Model
- Let's just spend a bit of time tying this into the model
- First some definitions.

# Definitions

- A **weighted finite-state automaton** adds probabilities to the arcs
  - The sum of the probabilities leaving any arc must sum to one
- A **Markov chain** is a special case of a WFST in which the input sequence uniquely determines which states the automaton will go through
- Markov chains can't represent inherently ambiguous problems
  - Useful for assigning probabilities to unambiguous sequences



# Hidden Markov Model

- A **Hidden Markov Model** is an extension of a Markov model in which the input symbols are not the same as the states.
- This means **we don't know which state we are in.**
- In HMM POS-tagging:
  - Input symbols: words
  - States: part of speech tags

# First: First-order observable Markov Model

- a set of states
  - $Q = q_1, q_2 \dots q_N$ ; the state at time  $t$  is  $q_t$
- Current state only depends on previous state

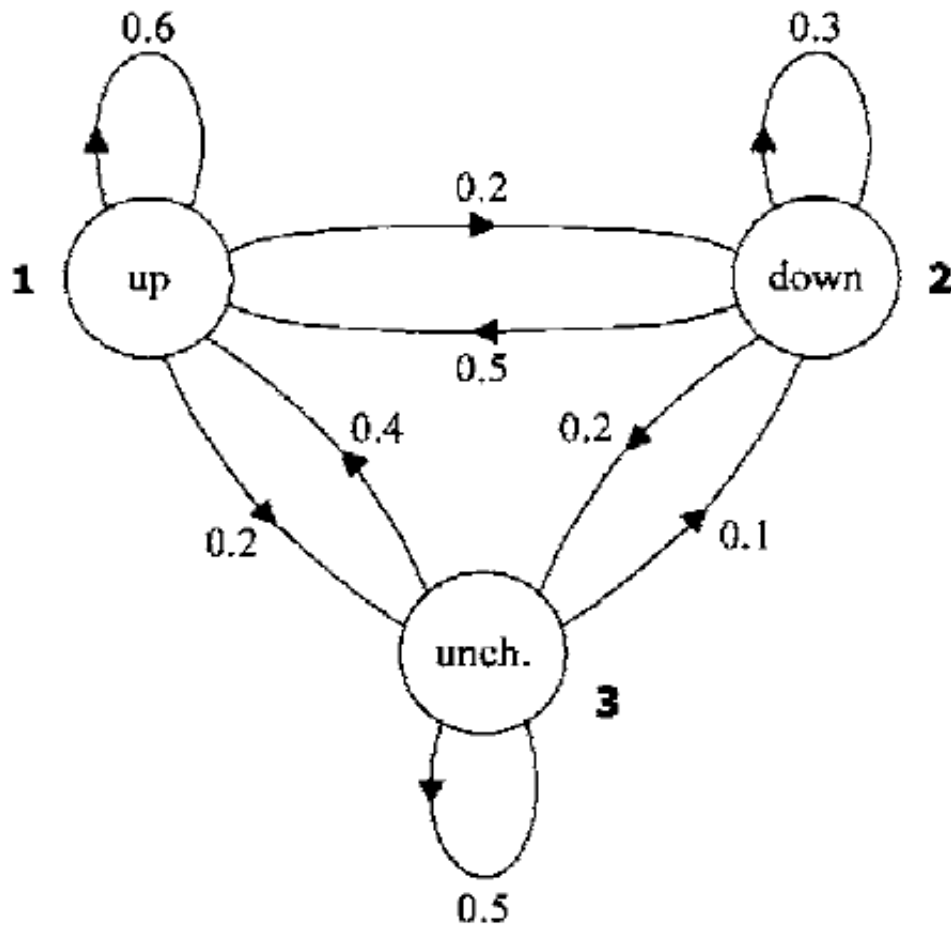
$$P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1})$$

- Transition probability matrix  $A$   
 $a_{ij} = P(q_t = j | q_{t-1} = i) \quad 1 \leq i, j \leq N$
- Special initial probability vector  $\pi$

- Constraints:  
 $\pi_i = P(q_1 = i) \quad 1 \leq i \leq N \quad \sum_{j=1}^N \pi_j = 1$

$$\sum_{j=1}^N a_{ij} = 1; \quad 1 \leq i \leq N$$

# Markov model for Dow Jones



Initial state probability matrix

$$\boldsymbol{\pi} = (\pi_i) = \begin{pmatrix} 0.5 \\ 0.2 \\ 0.3 \end{pmatrix}$$

State-transition probability matrix

$$\mathbf{A} = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

# Markov Model for Dow Jones

- What is the probability of 5 consecutive up days?
- Sequence is up-up-up-up-up
- I.e., state sequence is 1-1-1-1-1
- $P(1,1,1,1,1) =$   
 $\pi_1 a_{11} a_{11} a_{11} a_{11} = 0.5 \times (0.6)^4 = 0.0648$

# Hidden Markov Models

- a set of states
  - $Q = q_1, q_2 \dots q_N$ ; the state at time  $t$  is  $q_t$
- Transition probability matrix  $A = \{a_{ij}\}$   
 $a_{ij} = P(q_t = j | q_{t-1} = i) \quad 1 \leq i, j \leq N$
- Output probability matrix  $B = \{b_i(k)\}$   
 $b_i(k) = P(X_t = o_k | q_t = i)$
- Special initial probability vector  $\pi$   
 $\pi_i = P(q_1 = i) \quad 1 \leq i \leq N$
- Constraints:  
$$\sum_{j=1}^N a_{ij} = 1; \quad 1 \leq i \leq N \qquad \sum_{k=1}^M b_i(k) = 1 \qquad \sum_{j=1}^N \pi_j = 1$$

# Assumptions

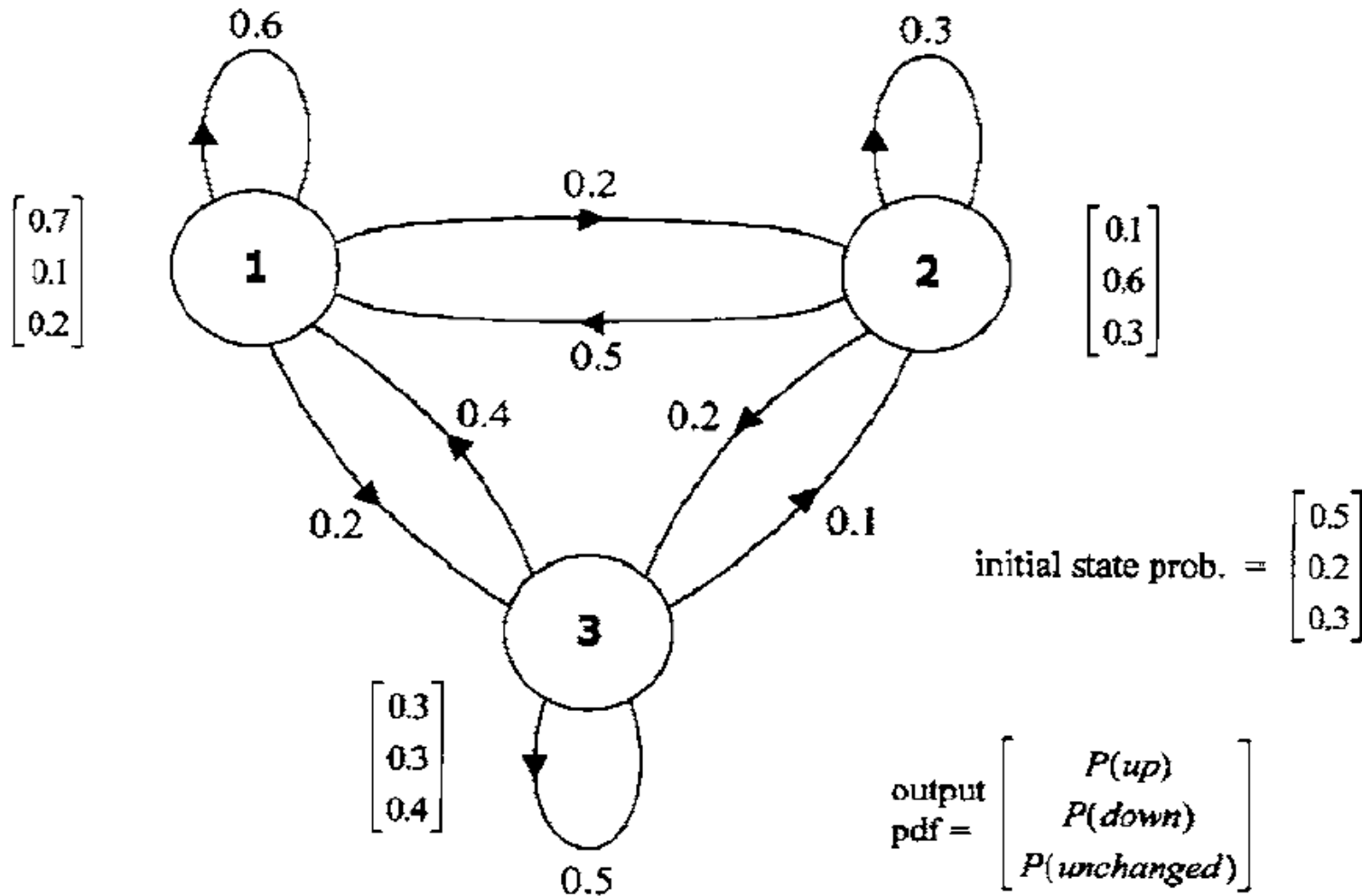
- Markov assumption:

$$P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1})$$

- Output-independence assumption

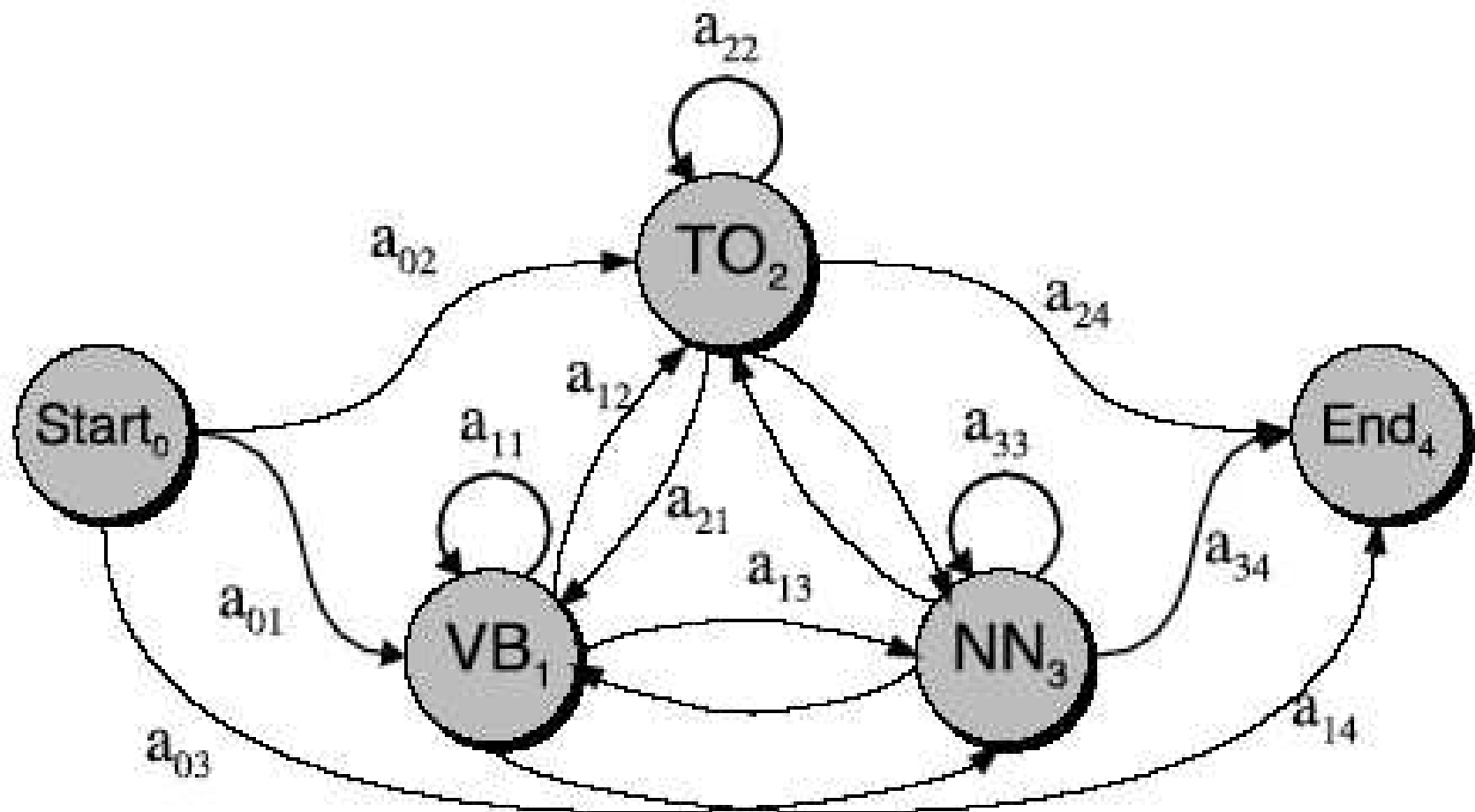
$$P(o_t | O_1^{t-1}, q_1^t) = P(o_t | q_t)$$

# HMM for Dow Jones



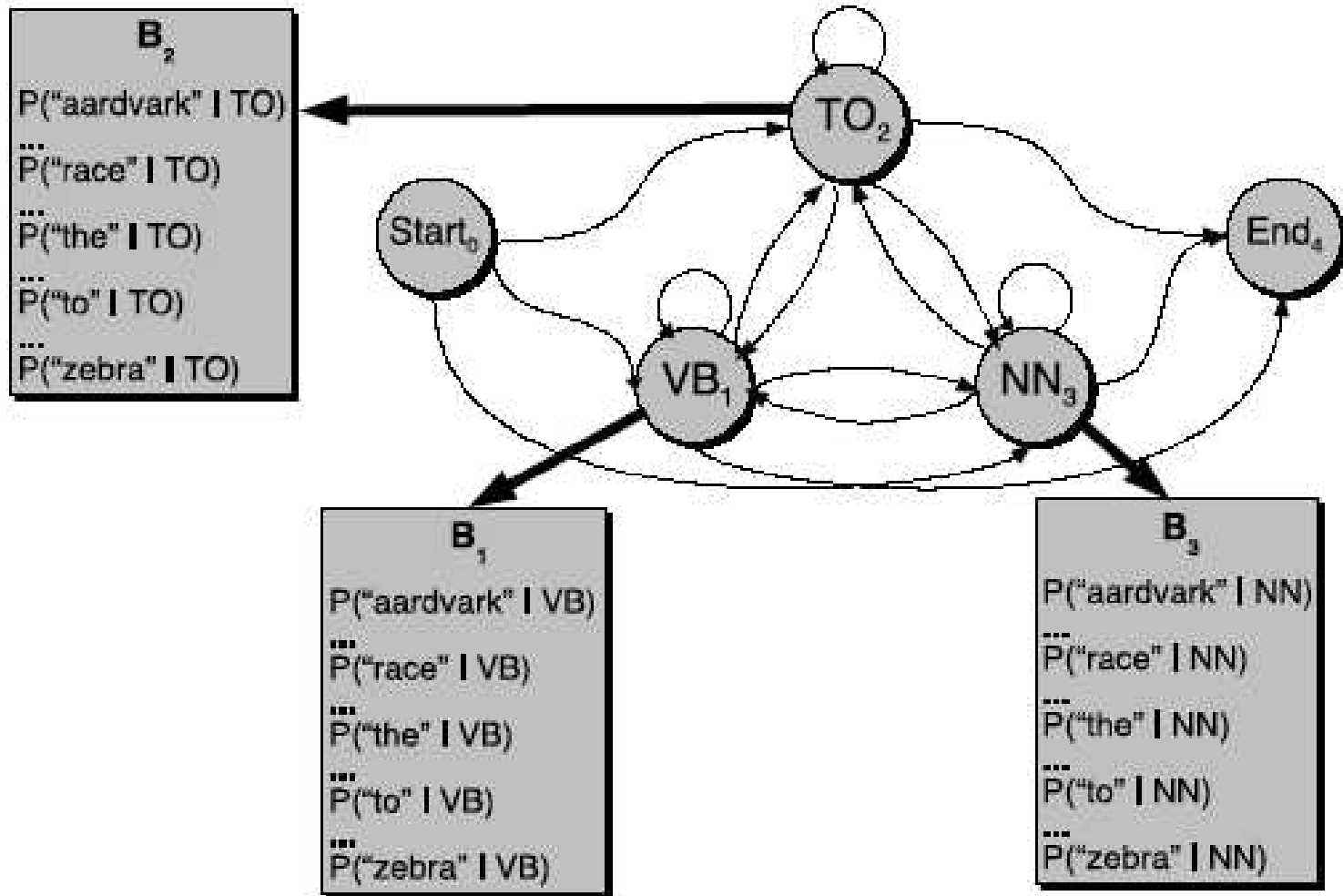
From Huang et al.

Weighted FSN corresponding to hidden states of HMM, showing A probs





# B observation likelihoods for POS HMM



# The A matrix for the POS HMM

	<b>VB</b>	<b>TO</b>	<b>NN</b>	<b>PPSS</b>
<b>&lt;s&gt;</b>	.019	.0043	.041	.067
<b>VB</b>	.0038	.035	.047	.0070
<b>TO</b>	.83	0	.00047	0
<b>NN</b>	.0040	.016	.087	.0045
<b>PPSS</b>	.23	.00079	.0012	.00014

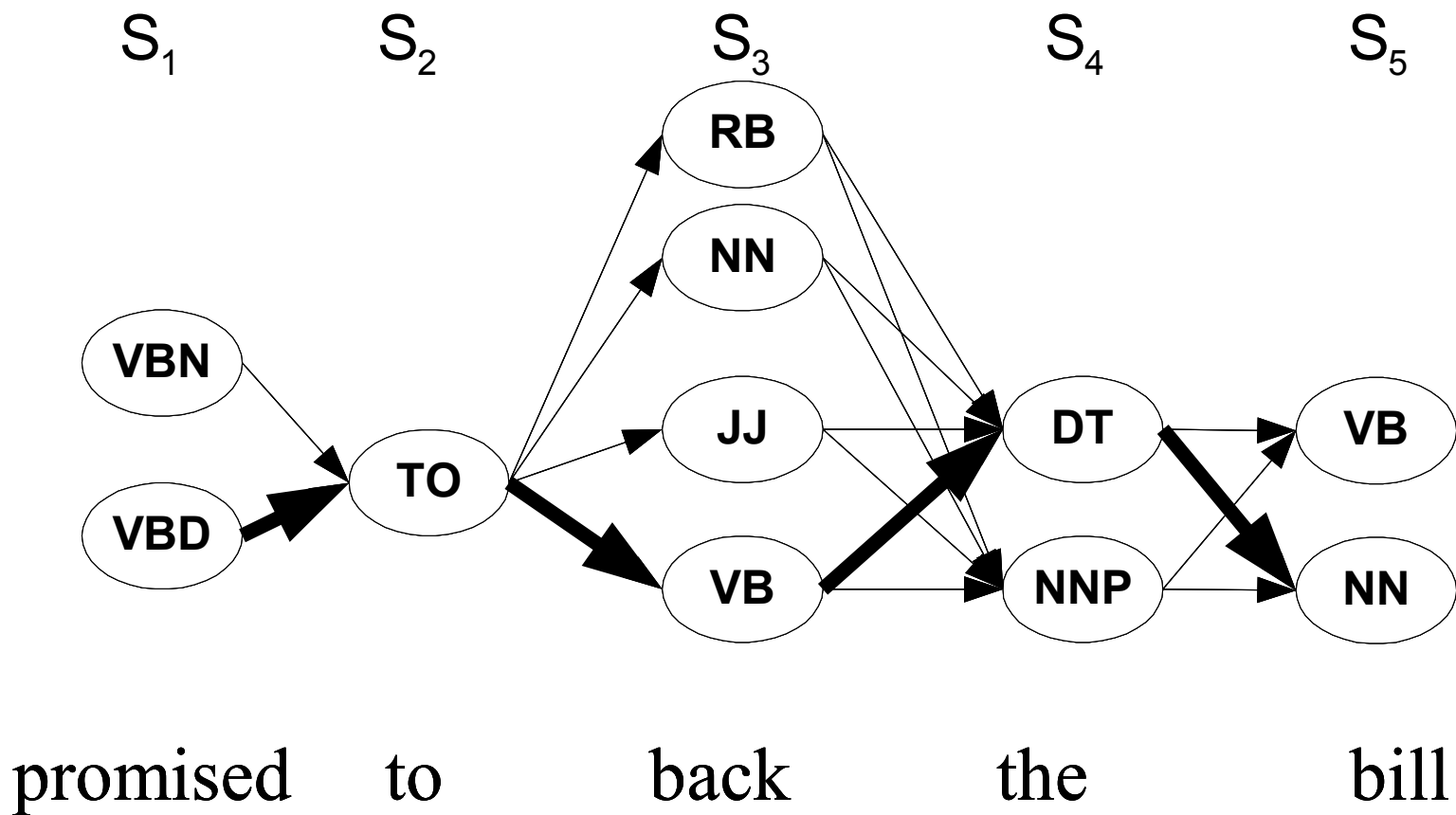
**Figure 4.15** Tag transition probabilities (the  $a$  array,  $p(t_i|t_{i-1})$ ) computed from the 87-tag Brown corpus without smoothing. The rows are labeled with the conditioning event; thus  $P(PPSS|VB)$  is .0070. The symbol  $\langle s \rangle$  is the start-of-sentence symbol.

# The B matrix for the POS HMM

	<b>I</b>	<b>want</b>	<b>to</b>	<b>race</b>
<b>VB</b>	0	.0093	0	.00012
<b>TO</b>	0	0	.99	0
<b>NN</b>	0	.000054	0	.00057
<b>PPSS</b>	.37	0	0	0

**Figure 4.16** Observation likelihoods (the *b* array) computed from the 87-tag Brown corpus without smoothing.

# Viterbi intuition: we are looking for the best 'path'



# The Viterbi Algorithm

**function** VITERBI(*observations* of len  $T$ , *state-graph*) **returns** *best-path*

$num\text{-}states \leftarrow \text{NUM-OF-STATES}(state\text{-}graph)$

Create a path probability matrix  $viterbi[num\text{-}states+2, T+2]$

$viterbi[0,0] \leftarrow 1.0$

**for** each time step  $t$  **from** 1 **to**  $T$  **do**

**for** each state  $s$  **from** 1 **to**  $num\text{-}states$  **do**

$viterbi[s,t] \leftarrow \max_{1 \leq s' \leq num\text{-}states} [viterbi[s',t-1] * a_{s',s}] * b_s(o_t)$

$back\text{-}pointer[s,t] \leftarrow \operatorname{argmax}_{1 \leq s' \leq num\text{-}states} [viterbi[s',t-1] * a_{s',s}] * b_s(o_t)$

Backtrace from highest probability state in final column of  $viterbi[ ]$  and return path

# Intuition

- The value in each cell is computed by taking the **MAX** over all paths that lead to this cell.
- An extension of a path from state  $i$  at time  $t-1$  is computed by multiplying:
  - Previous path probability from previous cell  $viterbi[t-1,i]$
  - Transition probability  $a_{ij}$  from previous state  $i$  to current state  $j$
  - Observation likelihood  $b_j(o_t)$  that current state  $j$  matches observation symbol  $t$

	<b>VB</b>	<b>TO</b>	<b>NN</b>	<b>PPSS</b>
<s>	.019	.0043	.041	.067
<b>VB</b>	.0038	.035	.047	.0070
<b>TO</b>	.83	0	.00047	0
<b>NN</b>	.0040	.016	.087	.0045
<b>PPSS</b>	.23	.00079	.0012	.00014

	<b>I</b>	<b>want</b>	<b>to</b>	<b>race</b>
<b>VB</b>	0	.0093	0	.00012
<b>TO</b>	0	0	.99	0
<b>NN</b>	0	.000054	0	.00057
<b>PPSS</b>	.37	0	0	0

# Viterbi example

5	end					
4	NN	$.041 * 1.0 * 0 = 0$				
3	TO	$.0042 * 1.0 * 0 = 0$				
2	VB	$.019 * 1.0 * 0 = 0$				
1	PPSS	$.067 * 1.0 * .37 = .025$				
0	start	1.0				

#            I            want            to            race            #  
 0            1            2            3            4            5

$\text{MAX}( 0 * .0040, 0 * .83, 0 * .0038, .025 * .23 ) = .025 * .23 = .0055$ . Then  $.0055 * .0093 = .000051$



# Tagging in other languages

- Idea:

- First do morphological parsing
- Get all possible parses
- Treat each parse for a word as a “POS tag”
- Use a tagger to disambiguate

1. Yerdeki **izin** temizlenmesi gerek. iz + Noun+A3sg+Pnon+Gen  
**The trace** on the floor should be cleaned.
  
2. Üzerinde parmak **izin** kalmış iz + Noun+A3sg+Pnon+Nom  
**Your finger print** is left on (it).
  
3. İçeri girmek için **izin** alman gerekiyor. izin + Noun+A3sg+Pnon+Nom  
You need a **permission** to enter.

# Error Analysis

- Look at a confusion matrix

	<b>IN</b>	<b>JJ</b>	<b>NN</b>	<b>NNP</b>	<b>RB</b>	<b>VBD</b>	<b>VBN</b>
<b>IN</b>	-	.2			.7		
<b>JJ</b>	.2	-	<b>3.3</b>	2.1	1.7	.2	<b>2.7</b>
<b>NN</b>		<b>8.7</b>	-				.2
<b>NNP</b>	.2	<b>3.3</b>	<b>4.1</b>	-	.2		
<b>RB</b>	<b>2.2</b>	2.0	.5		-		
<b>VBD</b>		.3	.5			-	<b>4.4</b>
<b>VBN</b>		<b>2.8</b>				<b>2.6</b>	-

- See what errors are causing problems
  - Noun (NN) vs ProperNoun (NN) vs Adj (JJ)
  - Adverb (RB) vs Particle (RP) vs Prep (IN)
  - Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)
- **ERROR ANALYSIS IS ESSENTIAL!!!**

# Summary

- Part of speech tagging
  - Parts of speech
  - What's POS tagging good for anyhow?
  - Tag sets
  - Rule-based tagging
  - Statistical tagging
    - Simple most-frequent-tag baseline
  - Important Ideas
    - Evaluation: % correct, training sets and test sets
    - Unknown words
    - Error analysis
  - TB tagging
  - HMM tagging